**DOKUZ EYLÜL UNIVERSITY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**E-BOOK ANALYSIS AND REPRESENTATION**

**Assignment Report**

**by**

**Mehmet Baran SELÇUK**

**January 2020**

**İZMİR**

Contents

[1 Introduction](#_Toc61092604) iii

[2 Methodology](#_Toc61092605) iii

[2.1 Structure of Your Project](#_Toc61092606) iii

[2.2 Encountered Problems and Solutions](#_Toc61092607) iv

[2.3 Improvements](#_Toc61092608) iv

[3 Experimentation](#_Toc61092609) v

4 Conclusion ……………………………………………………………………v

[Appendix A: Code](#_Toc61092610) vi

[Appendix B: Screenshots of your use cases](#_Toc61092611) xii

5 References……………………………………………………………………xv

# 

# Introduction

This program is an e-book analysis program that shows you the most common words in the book you choose. The program can show you connection and difference between two books by showing common and distinct words if you wish.

While programming the application , I used python that we had learned in the lab section and I utilize some libraries such as ‘requests’ and ‘beautifulsoup’ , also downloading e-books in the internet required a little bit html programming language knowledge.

# Methodology

## Structure of Your Project

Firstly, the structure of e-book analysis application consists libraries, functions, procedure and lots of if statement and for loops.

There are three libraries that I import the project. One of them is requests, the purpose of using requests module is taking the e-book’s data in the internet. The other module is BeautifulSoup and the goal is organizing e-book’s data that I got in the internet. By doing this, I have used html parser. The last library I used is operator, I used it for arrange the words and frequencies to the ordered pair.

Before getting the functions, there are some questions that user had to answer. How many books do he/she wants to examine? How many word frequency he/she wants to see? Then he/she writes book names and program does its job.

I determined to slice the codes for use easiness and make the code systematic by using functions and procedure. I had to write the book’s data to the text file and to do that I used procedure structure. Then, I create first function for reading the text file. It returns content of book and the content had to regulate stop-words, punctuations or numbers. The second function does that work and returns list structure. There is one last function except the main function and it uses for counting words numbers, set the order according to frequency and put them in the dictionary structure.

Lastly, program print common word/words and distinct words if user wants to examine two books by for loops and if statements.

## Encountered Problems and Solutions

When writing and reading text file there was an unexpected encoding problem, to solve that I encoded text file to utf-8 form.

Deleting stop-words and punctuations are more complicated I expected but it fixed by using for loops and if statements in function.

When analysis two books, the common words had to write with respect to sum of frequency and I find frequencies of each book’s words, sum them, finally write them in ordered form.

Finding distinct words is easier than finding common words but some typos and mistakes might occur in the end because of html parser method.

Some books have different url types and I had to prevent some typos. So that, I decided to get in the html data of the book site and get the full link of printable versions.

I used word counting program which I specify in references for making some controls of outputs but I realize some differences. The reason of that must be differences of used methods and stop-words that I take on the internet.

## Improvements

I used format method when print words to the screen and it provides more organized view.

By going in the html data of book site and get the full link of printable versions of books is prevents some errors.

# Experimentation

If user wants to learn about content of a specific book from wikibooks or wikisource, he/she understands by looking the common words.

* Planet Earth 🡪 <https://en.wikibooks.org/wiki/Planet_Earth> (EXECUTED CORRECTLY)
* Human Physiology 🡪 <https://en.wikibooks.org/wiki/Human_Physiology> (EXECUTED CORRECTLY)

If user wants to know how many words in the specific book from wikibooks or wikisource, he/she easily learn that.

* Algorithms 🡪 <https://en.wikibooks.org/wiki/Algorithms> (EXECUTED CORRECTLY)

If user wants to learn similarity of two books in wikibooks or wikisource, he/she learn that just looking the common and distinct words.

* Non Programmer’s Tutorial for Python 2.6 🡪 (EXECUTED-CORRECTLY) <https://en.wikibooks.org/wiki/Non_Programmer%27s_Tutorial_for_Python_2.6>
* Non Programmer’s Tutorial for Python 3 🡪 (EXECUTED-CORRECTLY)

<https://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_3>

Some frequency problems might be occur because of method and stop words I used.

# Conclusion

To sum up, I made an application which download the e-books from the internet and counts the common words and lists them or if there are two books in range, compare them and lists common and distinct words with respect to word frequencies. I have learned that importing some libraries and using them. Using functions and procedure more effectively, getting data from the internet by using html parse method. So that, I have used almost every information we learned from lab sections and consolidate them.

# Appendix A: Code

CODES OF PROJECT

import requests  
from bs4 import BeautifulSoup  
import operator  
  
  
# WRITING CONTENT TO THE TEXT FILE  
def writing\_file(content, book):  
 file = open(book+".txt", "w", encoding="utf-8")  
 file.write(content)  
 file.close()  
  
  
# READING CONTENT TO THE TEXT FILE  
def reading\_file(book\_name):  
 book = open(book\_name+".txt", "r", encoding="utf-8")  
 content = book.read()  
 book.close()  
 return content  
  
  
# REGULATES THE WORDS AND RETURNS SIMPLEST FORM OF WORDS  
def regulation(all\_words):  
 simple\_words = []  
 non\_punctuation = []  
 non\_numbers = []  
 punctuations = "∀æß~¨´`!'^+%&/()=?\_-\*|{}][{½$#£\"><@.⋅,;’:\\" + chr(775)  
 numbers = "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"  
 stop\_words = ["←", "name", "i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself",  
 "yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its",  
 "itself", "they", "them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this", "that",  
 "these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have", "has", "had",  
 "having", "do", "does", "did", "doing", "a", "an", "the", "and", "but", "if", "or", "because", "as",  
 "until", "while", "of", "at", "by", "for", "with", "about", "against", "between", "into", "through",  
 "during", "before", "after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off",  
 "over", "under", "again", "further", "then", "once", "here", "there", "when", "where", "why", "how",  
 "all", "any", "both", "each", "few", "more", "most", "other", "some", "such", "no", "nor", "not",  
 "only", "own", "same", "so", "than", "too", "very", "s", "t", "can", "will", "just", "don", "should",  
 "now", 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q',  
 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '.', '⋅', '∀']  
  
 # DELETING ALL PUNCTUATIONS  
 for punctuation in punctuations:  
 if punctuation in all\_words:  
 all\_words = all\_words.replace(punctuation, " ")  
 non\_punctuation = all\_words.split()  
  
 # DELETING ALL NUMBERS  
 for word in non\_punctuation:  
 for number in numbers:  
 if number in word:  
 word = word.replace(number, "")  
 if len(word) > 0:  
 non\_numbers.append(word)  
  
 # DELETING ALL STOPWORDS  
 for word in non\_numbers:  
 for stop\_word in stop\_words:  
 if word in stop\_word:  
 word = word.replace(stop\_word, "")  
 if len(word) > 0:  
 simple\_words.append(word)  
 return simple\_words  
  
  
# COUNTING WORD FREQUENCIES , REVERSE THEM IN THE DICTIONARY AND RETURN THE LIST FORM  
def freq\_dictionary(simple\_words):  
 word\_frequency = {}  
  
 for word in simple\_words:  
 if word in word\_frequency:  
 word\_frequency[word] += 1  
 else:  
 word\_frequency[word] = 1  
  
 sorted\_frequency\_1 = sorted(word\_frequency.items(), key=operator.itemgetter(1), reverse=True)  
 return sorted\_frequency\_1  
  
  
def main():  
 link2 = ""  
 book2 = ""  
 sorted\_frequency\_2 = []  
  
 # ASKING TO USER THAT HOW MANY BOOKS DOES HE\SHE WANTS TO EXAMINE  
 book\_number = int(input("How many books do you want to examine (1 or 2) : "))  
 link\_type = input("Which one do you prefer Wikibooks or Wikisource : ")  
  
 # ASKING TO USER WHETHER HE\SHE WANTS TO ENTER FREQUENCY OR NOT  
 answer = input("Do you want to enter a word frequency you wish to see ?(DEFAULT: 20) (Y or N) : ")  
  
 if answer == 'Y' or answer == 'y':  
 frequency = int(input("How many word frequency you wish to see : "))  
 else:  
 frequency = 20  
  
 # IF USER WANTS TO EXAMINE ONE BOOK  
 if book\_number == 1:  
 # GETTING THE BOOK NAME AND CREATING LINK VARIABLE  
 book1 = input("Please enter the book name : ")  
  
 if "'" in book1:  
 book1 = book1.replace("'", "%27")  
  
 # ASKING SOURCE WHICH DOWNLOAD THE BOOK  
 if link\_type == "Wikibooks" or "wikibooks":  
 link1 = ("https://en.wikibooks.org/wiki/" + book1)  
  
 # GETTING LINK OF PRINT VERSION OF BOOK FOR PREVENT SOME ERRORS  
 request = requests.get(link1)  
 soup = BeautifulSoup(request.content, "html.parser")  
 path = soup.find\_all("td", {  
 "style": "color:black; text-align:left; vertical-align:middle; padding:0.5em; padding-left:0em; border:none;"})  
  
 for link in path:  
 links = link.find("a")  
 p = links.get\_text()  
 if p == "printable version":  
 link\_version = links.get("href")  
 link1 = ("https://en.wikibooks.org" + link\_version)  
 break  
  
 else:  
 link1 = ("https://en.wikisource.org/wiki/" + book1)  
 # SOME CORRECTIONS  
 book1 = book1.replace("/", " ")  
 book1 = book1.replace("%27", "'")  
 book1 = book1.replace("\"", " ")  
  
 # IF USER WANTS TO EXAMINE TWO BOOKS  
 else:  
 # GETTING BOOK NAMES AND CREATING LINK VARIABLES  
 # BOOK1  
 book1 = input("Please enter the first book's name : ")  
  
 if "'" in book1:  
 book1 = book1.replace("'", "%27")  
  
 if link\_type == "Wikibooks" or "wikibooks":  
 link1 = ("https://en.wikibooks.org/wiki/" + book1)  
  
 # GETTING LINK OF PRINT VERSION OF BOOK FOR PREVENT SOME ERRORS  
 request = requests.get(link1)  
 soup = BeautifulSoup(request.content, "html.parser")  
 path = soup.find\_all("td", {  
 "style": "color:black; text-align:left; vertical-align:middle; padding:0.5em; padding-left:0em; border:none;"})  
  
 for link in path:  
 links = link.find("a")  
 p = links.get\_text()  
 if p == "printable version":  
 link\_version = links.get("href")  
 link1 = ("https://en.wikibooks.org" + link\_version)  
 break  
  
 else:  
 link1 = ("https://en.wikisource.org/wiki/" + book1)  
  
 # SOME CORRECTIONS  
 book1 = book1.replace("/", " ")  
 book1 = book1.replace("%27", "'")  
 book1 = book1.replace("\"", " ")  
  
 # BOOK2  
 book2 = input("Please enter the second book's name : ")  
  
 if "'" in book2:  
 book2 = book2.replace("'", "%27")  
  
 if link\_type == "Wikibooks" or "wikibooks":  
 link2 = ("https://en.wikibooks.org/wiki/" + book2)  
  
 # GETTING LINK OF PRINT VERSION OF BOOK FOR PREVENT SOME ERRORS  
 request = requests.get(link2)  
 soup = BeautifulSoup(request.content, "html.parser")  
 path = soup.find\_all("td", {  
 "style": "color:black; text-align:left; vertical-align:middle; padding:0.5em; padding-left:0em; border:none;"})  
  
 for link in path:  
 links = link.find("a")  
 p = links.get\_text()  
 if p == "printable version":  
 link\_version = links.get("href")  
 link2 = ("https://en.wikibooks.org" + link\_version)  
 break  
  
 else:  
 link2 = ("https://en.wikisource.org/wiki/" + book2)  
  
 # SOME CORRECTIONS  
 book2 = book2.replace("/", " ")  
 book2 = book2.replace("%27", "'")  
 book2 = book2.replace("\"", " ")  
  
 # GETTING DATA FROM THE LINK BY USING SOME MODULES  
 data1 = requests.get(link1)  
 beautiful\_data1 = BeautifulSoup(data1.content, "html.parser")  
  
 # GET DATA TO TEXT CONTENT  
 content1 = beautiful\_data1.find("body")  
 content1 = content1.text  
  
 # CALLING TO WRITE\_FILE FOR WRITE CONTENT TO THE TEXT FILE  
 writing\_file(content1, book1)  
  
 # CALLING TO READ FUNCTION FOR READING BOOK FROM THE TEXT FILE  
 read1 = reading\_file(book1)  
  
 # CALLING REGULATION FUNCTION AND SETTING THE APPROPRIATE FORM OF WORDS  
 all\_words1 = read1.lower()  
 simple\_words1 = regulation(all\_words1)  
  
 # CALLING FREQ\_DICTIONARY FUNCTION AND PRINT FREQUENCY OF WORDS  
 sorted\_frequency\_1 = freq\_dictionary(simple\_words1)  
  
 # IF THERE ARE 2 BOOKS DO THE SAME PROCESSES TO SECOND BOOK  
 if book\_number == 2:  
 data2 = requests.get(link2)  
 beautiful\_data2 = BeautifulSoup(data2.content, "html.parser")  
 content2 = beautiful\_data2.find("body")  
 content2 = content2.text  
 writing\_file(content2, book2)  
 read2 = reading\_file(book2)  
 all\_words2 = read2.lower()  
 simple\_words2 = regulation(all\_words2)  
 sorted\_frequency\_2 = freq\_dictionary(simple\_words2)  
  
 counter = 1  
 # FOR ONE BOOK  
 if book\_number == 1:  
 print()  
 print("BOOK 1 : ", book1, "\nCOMMON WORDS")  
 print(f"NO WORD FREQ\_1")  
 # FINDING FREQUENCIES OF WORD\WORDS AND SHOW THEM TO THE SCREEN  
 for i, j in sorted\_frequency\_1:  
 print(f"{counter:<3} {i:<12}{j:<5}")  
 counter += 1  
 # IF COUNTER REACHES FREQUENCY THAT USER WANTS, EXIT THE LOOP  
 if counter == frequency + 1:  
 break  
  
 # FOR TWO BOOKS  
 if book\_number == 2:  
 # FINDING SUM OF FREQUENCIES OF WORD\WORDS  
 freq\_sum = {}  
 for i, j in sorted\_frequency\_1:  
 for k, l in sorted\_frequency\_2:  
 if i == k:  
 # CREATING DICTIONARY ABOUT SUM OF FREQUENCIES  
 freq\_sum[i] = j+l  
  
 # SORT ITEMS AND CHANGE THE VARIABLE TYPE DICTIONARY TO LIST  
 freq\_sum\_list = sorted(freq\_sum.items(), key=operator.itemgetter(1), reverse=True)  
  
 # PRINT THE WORDS AND FREQUENCIES TO THE SCREEN  
 print()  
 print("BOOK 1 : ", book1, "\nBOOK 2 : ", book2, "\nCOMMON WORDS")  
 print(f"NO WORD FREQ\_1 FREQ\_2 FREQ\_SUM")  
 for m, n in freq\_sum\_list:  
 for i, j in sorted\_frequency\_1:  
 for k, l in sorted\_frequency\_2:  
 # PRINTING THE WORD THAT THE MOST REPEATED  
 if i == k and i == m:  
 print(f"{counter:<3} {i:<12}{j:<5} {l:<5} {n:<5}")  
 counter += 1  
 # IF COUNTER REACHES FREQUENCY THAT USER WANTS, EXIT THE LOOP  
 if counter == frequency+1:  
 break  
  
 print()  
 # FINDING DISTINCT WORDS FREQUENCIES FOR EACH BOOK AND SHOW THEM TO THE SCREEN  
 # FOR FIRST BOOK  
 counter = 1  
 print("BOOK 1 : ", book1, "\nDISTINCT WORDS")  
 print(f"NO WORD FREQ\_1")  
 for i, j in sorted\_frequency\_1:  
 distinct\_control = 0  
 for k, l in sorted\_frequency\_2:  
 if i == k:  
 break  
 else:  
 distinct\_control += 1  
 # IF CONTROL COUNT SAME AS SECOND BOOK'S WORD COUNT, IT MEANS THE WORD IS DISTINCT AND PRINT IT  
 if i != k and distinct\_control == len(sorted\_frequency\_2):  
 print(f"{counter:<3} {i:<12} {j:<5}")  
 counter += 1  
 # IF COUNTER REACHES FREQUENCY THAT USER WANTS, EXIT THE LOOP  
 if counter == frequency+1:  
 break  
  
 print()  
 # FOR SECOND BOOK  
 counter = 1  
 print("BOOK 2 : ", book2, "\nDISTINCT WORDS")  
 print(f"NO WORD FREQ\_2")  
 for i, j in sorted\_frequency\_2:  
 distinct\_control = 0  
 for k, l in sorted\_frequency\_1:  
 if i == k:  
 break  
 else:  
 distinct\_control += 1  
 # IF CONTROL COUNT SAME AS SECOND BOOK'S WORD COUNT, IT MEANS THE WORD IS DISTINCT AND PRINT IT  
 if i != k and distinct\_control == len(sorted\_frequency\_1):  
 print(f"{counter:<3} {i:<12} {j:<5}")  
 counter += 1  
 # IF COUNTER REACHES FREQUENCY THAT USER WANTS, EXIT THE LOOP  
 if counter == frequency + 1:  
 break  
  
  
main()

# Appendix B: Screenshots of your use cases

LOOKING FOR SIMILARITY OF TWO BOOKS

* Common Words

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Distinct Words

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

LOOKING FOR WORD COUNT OF A BOOK

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* CONTINUED and find word count.

metin, cihaz, metre içeren bir resim

Açıklama otomatik olarak oluşturuldu

LOOKING FOR CONTENT OF A BOOK

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* We can guess content of a book just looking to the common words.

# REFERENCES

Words Counting from : https://countwordsfree.com/

Learning format method from : https://blalyasar.blogspot.com/2019/12/python-print-fstrings-format-ve-s-neden.html

https://pyformat.info/

NLTK stop words from : https://gist.github.com/sebleier/554280

Python knowledge : https://docs.python.org/3/tutorial/datastructures.html

Books from : <https://en.wikibooks.org/wiki/Planet_Earth>

<https://en.wikibooks.org/wiki/Algorithms>

<https://en.wikibooks.org/wiki/Human_Physiology>

<https://en.wikibooks.org/wiki/Non_Programmer%27s_Tutorial_for_Python_2.6>

https://en.wikibooks.org/wiki/Non-Programmer%27s\_Tutorial\_for\_Python\_3

https://en.wikibooks.org/wiki/Main\_Page

Downloading and using pycharm from : https://www.jetbrains.com/pycharm/